Machine Comprehension

Using Bidirectional LSTM and Attention Mechanism

Presented By:
Aman Kumar, Asok Kalidass, Balaji D, Viswanath MS

DALHOUSIE UNIVERSITY

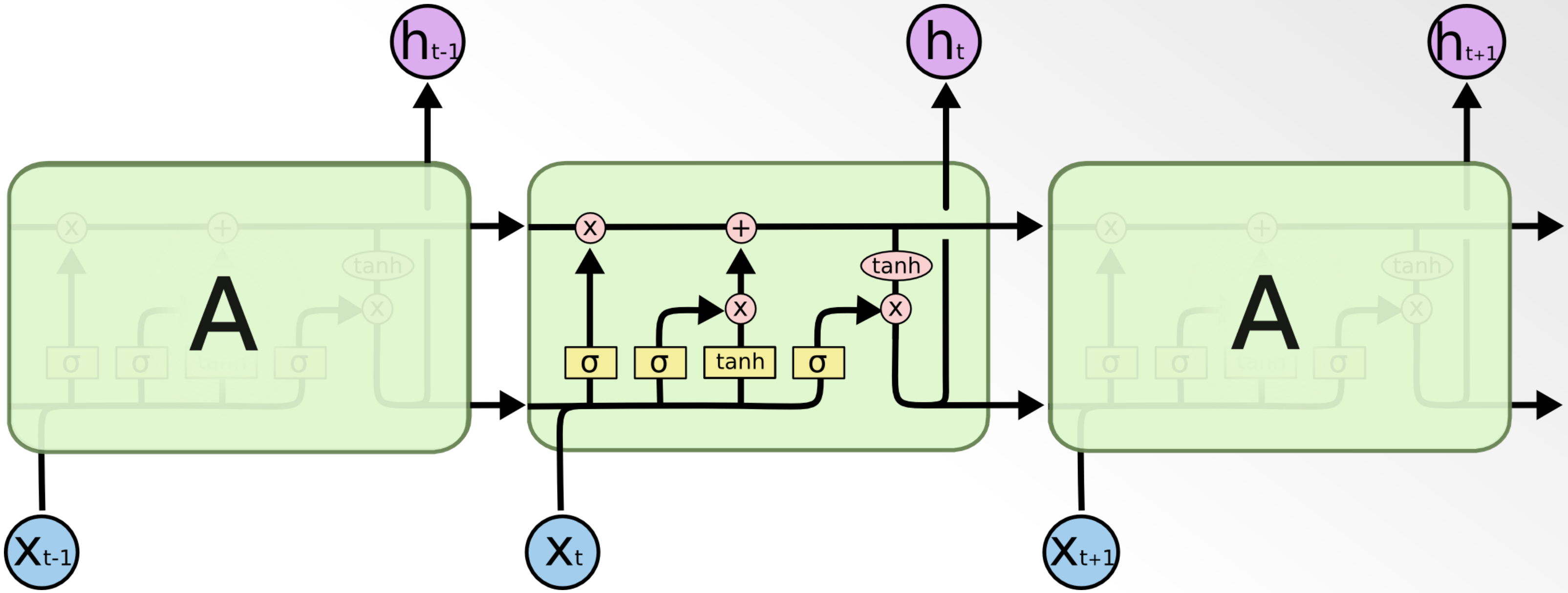
Machine Comprehension

- Comprehension of written language by machines.
- Textual Entailment Problem.
- Information Retrieval task.
- More Semantic high level interpretation required.

LSTM for Language Modelling

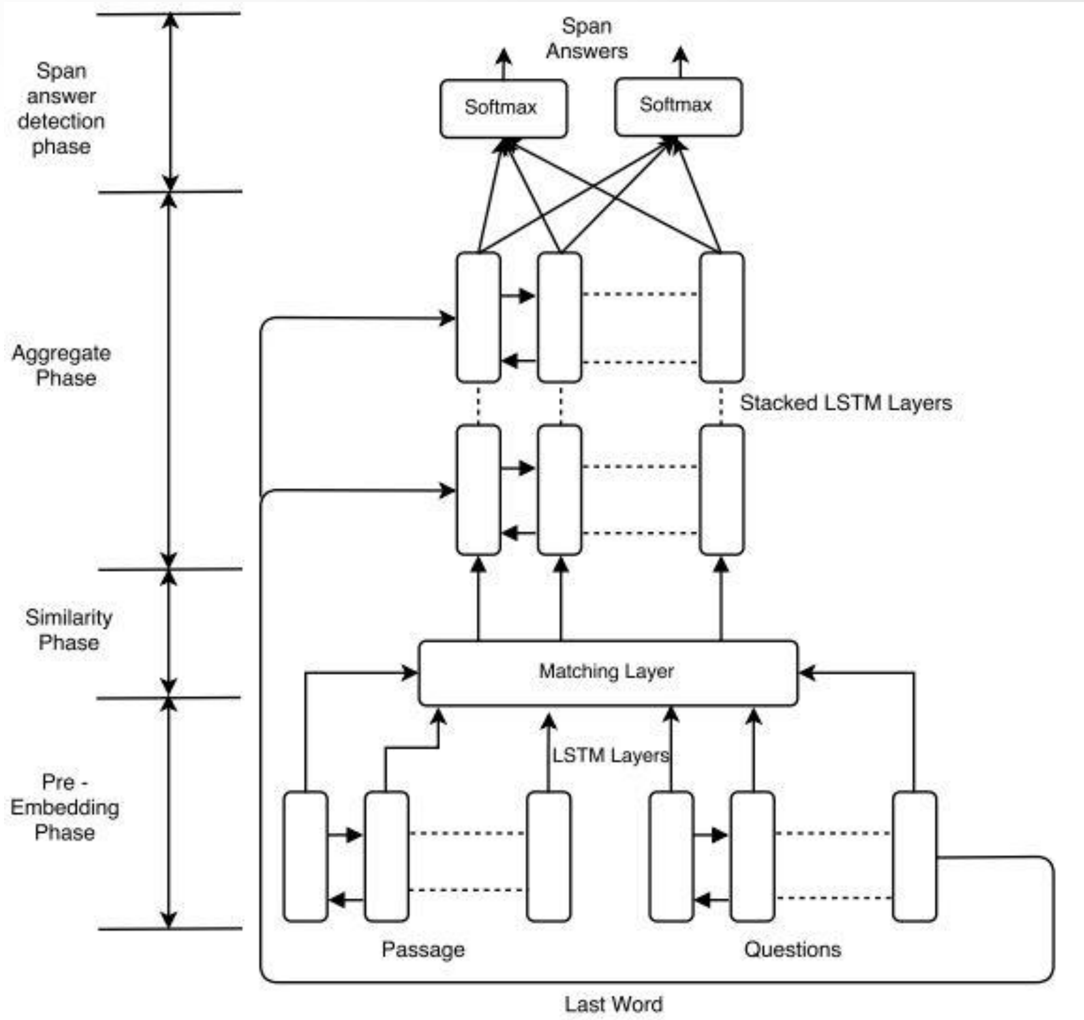
- Passage and Word have longer word sequences.
- Vanishing Gradient problem.
- LSTM (Schmidhuber et al.1997)
    - Sigmoid gates.
- Bidirectional LSTM to infer the language from both directions.
    - More knowledge interpretation in earlier stages.

LSTM Model



* http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Architecture



Layer Explanation

- Pre Embedding for Passage and Query words
  - Passage and Query words are converted to word x 300 dimensions using GloVe embedding.
- Bidirectional LSTM
  - Separate Bidirectional LSTMs applied for passage and query words.

Attention Layer

- Outputs of the LSTM Layers of both Passage and Query words are matched using cosine similarity.
- Max Pooling

$$\overleftarrow{f}_i = max_j \overleftarrow{HP_i} * \overleftarrow{HQ_j}$$

$$\vec{f_i} = max_j H\vec{P_i} * H\vec{Q_j}$$

Where,

HP,HQ - output vectors of Passage and Question LSTM Layers.

Attention Layer

- Average Pooling:

$$\overleftarrow{f}_i = avg_j \overleftarrow{HP_i} * \overleftarrow{HQ_j}$$

$$\vec{f}_i = avg_j H\vec{P}_i * H\vec{Q}_j$$

- Final Layer:
  - Outputs of Attention Layers are processed as inputs to the stacked Bidirectional LSTMs for interpreting high level representation.

Attention Layer

- Softmax Layers:
  - Two Softmax layers applied to outputs of stacked Bidirectional LSTM with dense of 1 class.
  - Argmax applied to find the start and end indices of the words in the passage.
- Span Selection:
  - The span of answer is generated based on the location of words in passage between start and end indices.

Related Works

- SQuAD uses Ngram, dependency tree and various other attributes.
- Shouhang Wang et al. Used LSTM and answer pointer to predict the fixed length span answers.
- Mingoon Seo used Bidirectional LSTM and attention mechanism based on context to query and query to context scheme.

Motivation

- Too much information on the web and all over the globe.
- Less time to parse it and often difficult to comprehend it.
- Acute impact in the field of medical, technical and legal documents.

RoadMap

- Embedding Layer
- LSTM Layer
- Attention Layer
- Output Layer

Embedding Layer

- SQuAD Dataset
- Glove Vector

SQuAD

The Stanford Question Answering Dataset

* https://twitter.com/pranavrajpurkar

SQuAD DataSet [1]

- Stanford Question Answering Dataset

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under **gravity**. The main forms of precipitation include drizzle, rain, sleet, snow, **graupel** and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals **within a cloud**. Short, intense periods of rain in scattered locations are called "showers".

What causes precipitation to fall?
gravity

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

Where do water droplets collide with ice crystals to form precipitation?
within a cloud

* http://wenchenli.github.io/2017/01/SQuAD

What is SQuAD?

- SQuAD is a reading comprehension dataset, containing number of questions given by crowd workers on a number of Wikipedia articles.
- Large quantity and high quality dataset.

Cons of other datasets

Datasets Available:

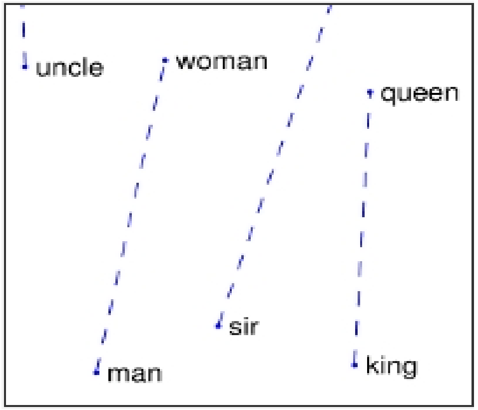
- MCTest, Algebra, WikiQA,TREC-QA, CNN/Daily Mail.
- Although real and difficult, are too small to support very expressive statistical models.
- Few dataset tasks is sentence selection, while ours requires selecting a specific span in the sentence.

Why SQuAD is different?

- In SQuAD dataset, answers include non-entities and can be much longer than single word while in other datasets, answers to questions are single words.
- High Quality and large dataset.

Glove: Global Vectors for Word Representation

- GloVe: It is an unsupervised learning algorithm for obtaining vector representations for words.
- Aggregated global word-word co-occurrence statistics from a corpus.
- Resulting representations showcase interesting linear substructures of the word vector space.



* https://nlp.stanford.edu/projects/glove/

Nearest Neighbor

- The Euclidean distance between 2 word vectors provides an effective method for measuring the linguistic or semantic similarity of the corresponding words.
- For Example, the nearest words related to Frogs are rana, litoria, load and eleutherodactylus etc.

Background – Neural Network

- We humans don't start new thinking from scratch for every word.
- We understand each word based on the previous knowledge.
- This behavior is imitated using the recurrent neural network.
- The feedback loops retain the past information.
- Consider the example,

The Sun rises in the *East.*

- The word East depend on the previous words Sun and Rises.

Background – Neural Network

Example 2:

I was born in India. I inherited citizenship of *India.*

- *The last word depends on the one of the previous word India.*
- *In example 2, the previous dependent word is farther than in example 1.*
- *As the gap grows, RNN* was unable to track the past information.*

**RNN – Recurrent Neural Network*

LSTM*

- The issue of retaining past information is overcome by LSTM (uses cell state to store information).

  Now consider examples [2],

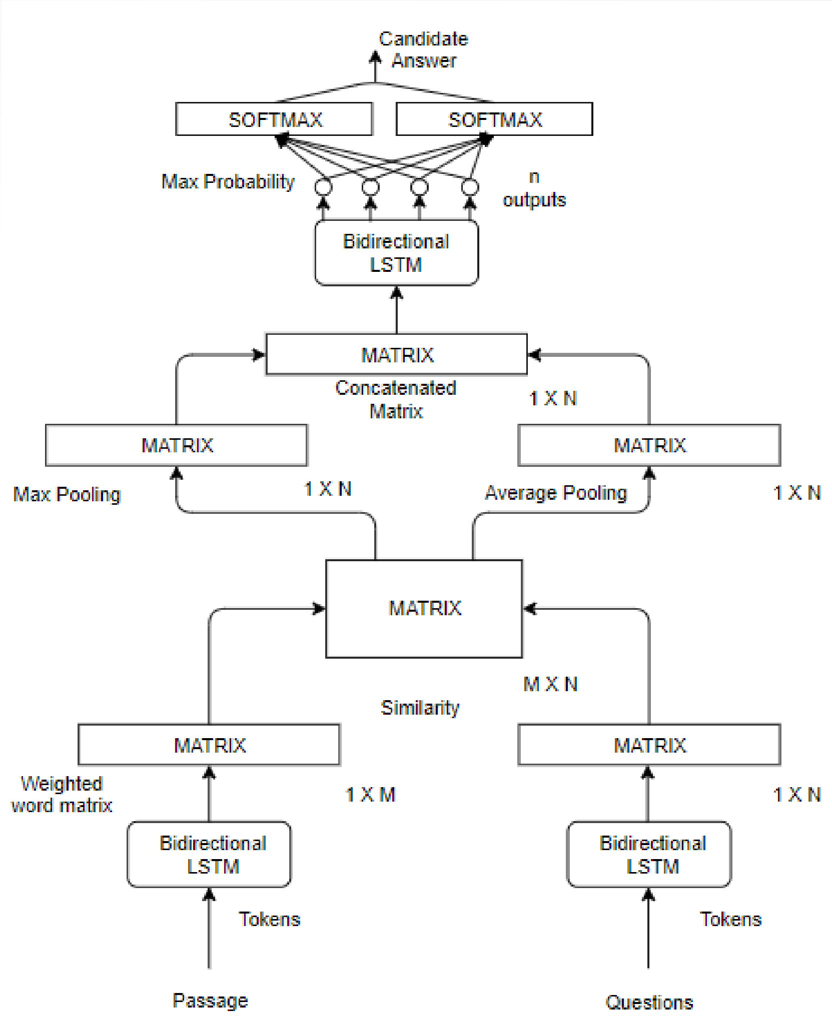
  He said, "Teddy bears are on sale"

  He said, "Teddy Roosevelt was a great president"

- The word Teddy refers to either Teddy bear or the president based on the future word.
- Here comes the necessity for a Bidirectional LSTM.
- They train on as is input sequence than on its reverse copy.

**LSTM – Long Short Term Memory*

Implementation – Flow Diagram



Implementation

* https://blog.keras.io/keras-as-a-simplified-interface-to-tensorflow-tutorial.html

Why Keras? [3]

- High level Neural Networks API.
- Enables fast implementation.
- User friendly, modular and extensible.
- Runs seamlessly on CPU & GPU.

Challenges Faced

- Unicode encoding
  - \u002D – Unicode for ‘-’.
  - Could not be used as dictionary index.

- Computational Resources
  - CPU vs GPU.
  - NVIDIA Tesla P100 .

- Dataset Size
  - Minimum Size of SQuAD : 5 MB.
  - Minimum Size of GloVe: 163 MB.

SQuAD Dataset Details

- Stored as a JSON


```
{
    "data":[
        {
            "title":"Super_Bowl_50",
            "paragraphs":[
                {
                    "context":"Super Bowl 50 was an American football game to determine
themed initiatives, as well as temporarily suspending the tradition of naming each
                    "qas":[
                        {
                            "answers":[
                                {
                                    "answer_start":177,
                                    "text":"Denver Broncos"
                                },
```

GloVe Details

- The 0.418 0.24968 -0.41242 0.1217 0.34527 - 0.044457 - 0.49688 ….
- It is in the form of a word followed by a vector of a fixed dimension.

Loading GloVe values

- Read each line from GloVe file.
- Split using space.
- We get 101 Values.
- First is a word all others represent the vector representation for the word.

```
def get_glove_values():
  with codecs.open('D:/glove.6B.100d.txt','rb',encoding='utf-8') as f:
      print("========> Importing GloVe Vector  <=========")
      #Parsing GloVe dataset
      for line in f:
          values = line.split()
          word = values[0].lower()
          embed_index[word] = np.asarray(values[1:], dtype='float32')
      print("Number of vectors loaded : "+ index))
```

Reading Data From SQuAD

- Open the SQuAD dataset.
- Obtain the first context and questions on the context.

```
with codecs.open('D:/Dev.json','rb',encoding='utf-8') as data_file:
    data = json.load(data_file)
    data_entity_list = []
    count_iterations = 0
    question='';
    for item in data['data']:
      paragraph = item['paragraphs']
      for ctx in paragraph:
        if count_iterations < CONST_INPUT_SIZE:
           context = ctx['context']
           vocab_context = tokenize_and_embed_words(context,True);
```

Word Embedding Phase

- Paragraph is split into words.
- For each word a vector is obtained from the GloVe.
- A matrix is created using these values.

```
words_in_input = text_to_word_sequence(inputValue)
count_words_input = 0
if isContext :
  vocab_context = np.zeros((len(words_in_input), 100))
  for word in words_in_input:
    if word in embed_index.keys():
      vocab_context[count_words_input] = embed_index[word]
      count_words_input += 1
  actual_data = vocab_context[0:count_words_input]
```

Applying Bidirectional LSTM

- Convert the matrix from previous step to tensor.
- Pass this this tensor to a layer of Bidirectional LSTM.

```
vocab_context_tensor = tensor.convert_to_tensor(vocab_context,dtype=tensor.float32)
embed_context = Embedding(input_dim=size_context, output_dim=100,input_length=size_context,trainable=False)
tensor_context = embed_context(vocab_context_tensor)
lstm_context = Bidirectional(LSTM(100))
lstm_context_out = lstm_context(tensor_context)
print(lstm_context_out)
#drop_1 = Dropout(0.5)(lstm_out)
```

Pending Implementation

- Cosine Similarity.
- A stack of LSTMs to derive the span of answers.

References

- [1] Rajpurkar, Pranav, Zhang, Jian, Konstantin, Liang, and Percy, “SQuAD: 100,000 Questions for Machine Comprehension of Text,” *[1606.05250] SQuAD: 100,000 Questions for Machine Comprehension of Text*, 11-Oct-2016. [Online]. Available: https://arxiv.org/abs/1606.05250. [Accessed: 05-Apr-2018].
- [2]“Sequence Models,” *Coursera*. [Online]. Available: https://www.coursera.org/learn/nlp-sequence-models. [Accessed: 05-Apr-2018].
- [3] “Keras: The Python Deep Learning Library.” *Keras Documentation*, keras.io/ [Accessed: 05-Apr-2018].